

Chapter 2

LITERATURE STUDY

2.1 OVERVIEW AND RELATED WORK

As the costs of high-end computer equipment get less the networking community is moving away from the traditional server-client and autonomous networking models. The sharing of resources, as a result of distributed computing, has changed the nature of network applications. Information is no longer housed at a central secured location, therefore network security has become the most important aspect in information networks today. Bandwidth requirements, the range of services and QoS are all necessary requirements for a functional network but a security failure is the most catastrophic and almost tend to negate the other requirements.

Field area networks are rapidly expanding to include a wide range of applications [1]. Embedded intelligent nodes on the network can be installed in a small to medium geographical area (e.g. an office, a group of flats or a factory) to monitor and control real-time processes. Such nodes are generally connected to a centralized gateway used by a service provider to monitor and control various applications. For such applications the client would not only require measurement, control and communication functionality but also a high level of security. The node must therefore be able to capture, manipulate and secure data and send it to the gateway, from where it is presented to the service provider. Field area and automation networks have mainly been used as stand-alone systems. With these technologies gaining in popularity and being used in new applications they become more distributed. This introduces new challenges and considerations in terms of security. In the power measurement/billing application the person with access to the node is not the owner but in fact the client. In addition the network runs through an entire apartment complex with a few nodes per residence. In this case the residents would be able to change either their own or their neighbours' readings.

The growth in popularity of ubiquitous computing requires the use of embedded network processors in everyday objects. Even though the idea of interaction between the digital devices around us could bring a great deal of convenience it also introduces great risks [9]. In the past much emphasis has been put on information security for high performance computers and data networks. These methods suffice for information systems incorporating powerful processing capabilities. The embedded and real-time nature of the network itself poses problems other than encountered in the networked PC environment. Nodes could be weak in storage or processing capability due to size or power concerns with real-

time operation placing constraints on the time that can be allocated to security overhead. As a result security technology used in the PC/Internet environment or developed with TCP/IP in mind will not be practical. The traditional security model identifies three main classes [10]:

- Confidentiality: Information not divulged to wrongful entity.
- Integrity: Information cannot be maliciously modified.
- Availability: The system cannot be prevented from performing its functions.

These classes are however all dependant on identification, authentication and authorization principles to be successful. A failure to authenticate correctly will easily lead to breaches in the rest of the security policy. Smart cards offer security mechanisms to provides security services under various circumstances [11], [12].

Entities from different vendors, or with different owners may be connected to the same network and therefore networks are no longer guaranteed to be autonomous and secure. Each entity needs a secure way to relate to its owner, vendor and peers in the network without being compromised. For distributed networks to be a commercially viable proposition a range of security services must be provided in order to protect the system from wrongful activity. Smart cards offer a simple, inexpensive method of incorporating a cryptographic processor into an embedded system. Each smartcard has a cryptographic processor that performs the algorithms used to lock, unlock and verify data. The smart card is also strongly protected against different security attacks [13]. These two technologies together could very much benefit everyday applications [5].

After studying the available literature it would appear that little or no work has been done on securing field area networks. This research dissertation will be one of the first to document information security principles and performance for embedded field area networks. An abundance of information is available on security methodology, smart card technology and distributed field area networks.

2.2 INFORMATION SECURITY

Information security is a field of study that is enjoying tremendous attention in computing communities. It can be described as an ever-growing technology and many organizations

are investing heavily to develop security features. There are three definable aspects to a secure system [14]:

- Threats: A potential for violation of security, which exists when there is a circumstance capability, action, or event that could breach security and cause harm. A security attack is a threat that has realised.
- Security mechanisms: An action that is designed to detect, prevent, or recover from a security attack.
- Security services: A service that enhances the security of the data processing systems using one or more security mechanisms.

Some of the most important goals of information security are listed below:

- Confidentiality: Confidentiality is the protection of transmitted data from attacks. This requires that an attacker cannot gain any useful information from analysis of the communication channels.
- Integrity: This deals with the assurance that data sent is the same as data received. No modification or removal of message data has taken place.
- Availability: An entity is not permitted to keep the computing and or communication resources from being of constructive use.
- Authentication: This assures the identities of the two communicating entities to be authentic (each is the entity it claims to be) and secondly it assures that the connection is not interfered with in such a manner that a third party can masquerade as one of the two legitimate parties for the purpose of unauthorized transmission or reception.

Possible services, mechanisms and threats are mentioned that would generally appear in most secure systems. These are explained in the ITU-T X.800 [15] and ISO 7498-2 [16] guidelines. A processing or communications service is provided by a system to give a specific kind of protection to system resources. Security policies are implemented by security services that in turn is implemented by security mechanisms. A more detailed discussion on security threats, mechanisms and services will follow later in this section.

In addition to the principles of information security there is still the question of security assurance. A sound method must be followed in formulating the security policy. This

ensures that all issues are considered and addressed. Guidelines for the management of security policies are described in the ISO 13335 [17] and ISO17799/BS7799 [18] standards. These standards describe a design method to follow to provide security assurance. There are also some organizations that audit and test the technical aspects of the system according to a set standard. An evaluation certificate is then issued stating the evaluation assurance level of the policy. A more detailed discussion on security management, risk analysis, compliance checking and assurance will follow later in the section.

2.2.1 Security Mechanisms

Security mechanisms are used to provide security services and guard against specific forms of attack [19]. Mechanisms can be divided into two classes: Specific security mechanisms and pervasive security mechanisms. Eight specific security mechanisms are defined for use in specific security services.

- Encipherment: The use of a mathematical algorithm to transform data into a form that is not readily intelligible. The transformation and subsequent recovery of the data depend on an algorithm and some encryption keys. This helps in authentication and key management techniques and provides both data and traffic flow confidentiality.
- Digital signature: Data appended to, or cryptographic transformation of, a data unit that allows the recipient of the data to prove the source and integrity of the data unit. It consists of a signing and a verification procedure and can be used to provide non-repudiation, origin authentication and/or integrity services and also plays a part in entity authentication.
- Access control mechanisms: A variety of mechanisms that enforces access rights to resources. These include access control lists and security labels.
- Data integrity mechanisms: Mechanisms used to assure the integrity of a data unit or stream of data units. One type can be used to provide both data origin authentication and data integrity. Mechanisms including sequence numbers and time stamps help detect replays of single data units and manipulation of a sequence of data units.
- Authentication exchanges: A mechanism intended to ensure the identity of an entity by means of information exchange. It consists of a series of cryptographic messages

exchanged between a pair of communicating entities.

- Traffic padding: The addition of data to conceal real volumes of data traffic. It is effective for providing traffic flow confidentiality if used in conjunction with encipherment.
- Routing control: Enables selection of particular physically secure routes for certain data and allows routing changes, especially when a breach of security is suspected.
- Notarisation: Integrity, origin and destination of data can be guaranteed by using a 3rd party notary. It can provide non-repudiation services.

Five pervasive security mechanisms are defined which describe functions that are not related to a specific service in particular.

- Trusted functionality: That which is perceived to be correct with respect to some criteria e.g. as established by a security policy.
- Security labels: The marking bound to a resource that names or designates the security attributes of that resource.
- Event detection: Detection of security-relevant events.
- Security audit trail: Data collected and potentially used to facilitate a security audit, which is an independent review and examination of system records and activities.
- Security recovery: Deals with requests from mechanisms, such as event handling and management functions, and takes recovery actions.

The relationship between security services and security mechanisms is shown in figure 2.1

2.2.2 Security Threats

Security threats can be divided into two categories [14]. A passive attack attempts to learn or make use of information from the system but does not affect system resources. An active attack attempts to alter resources or affect their operation. Passive attacks are difficult to detect but are easily prevented. On the other hand, it is quite difficult to prevent active attacks absolutely. Instead the goal is to detect them and to recover from any disruption or delays caused by them. Passive attacks take two forms:

Service	Encipherment	Digital Signature	Access control
Entity authentication	X	X	
Origin authentication	X	X	
Access control			X
Connection confidentiality	X		
Connectionless confidentiality	X		
Selective-field confidentiality	X		
Traffic flow confidentiality	X		
Connection integrity with recovery	X		
Connection integrity without recovery	X		
Selective field connection integrity	X		
Connectionless integrity	X	X	
Selective field connectionless integrity	X	X	
Non-repudiation of origin		X	
Non-repudiation of delivery		X	

Service	Authentication Exchange	Traffic padding	Routing control
Entity authentication	X		
Origin authentication			
Access control			
Connection confidentiality			X
Connectionless confidentiality			X
Selective-field confidentiality			
Traffic flow confidentiality		X	
Connection integrity with recovery			
Connection integrity without recovery			
Selective field connection integrity			
Connectionless integrity			
Selective field connectionless integrity			
Non-repudiation of origin			

Figure 2.1:
Service/mechanism correlation [14]

- Release of message contents (interception): An attack on confidentiality by trying to obtain the contents of a secret message.
- Traffic analysis: Opponents can determine the location and identity of communicating hosts. By observing the frequency and length of messages being exchanged the nature of communication taking place might be guessed.

Active attacks involve modification of the data stream or the creation of a false stream and can be divided into four categories:

- Masquerading: One entity pretends to be another entity in order to gain access to unauthorized resources. This is an attack on authentication.
- Replay: This involves the passive capture of a data unit and its subsequent retransmission to produce an unauthorized effect. This is an attack on authentication.
- Modification: A portion of a legitimate message is altered to produce an unauthorized effect.
- Denial of service: This attack prevents or inhibits the normal use or management of communication facilities. An asset of the system is destroyed or becomes unavailable or unusable.

Some threats to specific services are shown in figure 2.2. A more detailed discussion on security threats, mechanisms and services will follow later in this section.

2.2.3 Security Services

Services can be grouped into six separate categories: Authentication, access control, confidentiality, integrity, non-repudiation and availability.

2.2.3.1 Authentication

The authentication service is concerned with assuring that a communicating entity is the one it claims to be. Two specific authentication services are defined:

- Peer entity authentication: Provides for the corroboration of the identity of a peer entity in an association. It is provided during the establishment of a connection or at any other time during the data transfer phase. It attempts to provide confidence that an entity is not attempting either a masquerade or an unauthorized replay of a previous connection.

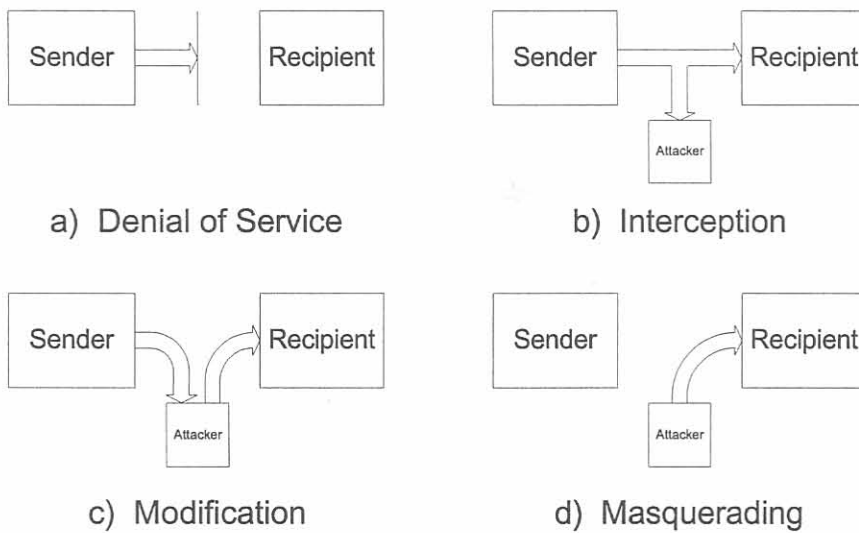


Figure 2.2:
a) Availability b) Confidentiality c) Integrity d) Authentication

- Data-origin authentication: Provides for the corroboration of the source of a data unit. Provides authentication where there were no previous interactions between communicating entities. This does not provide protection against modification or duplication attacks.

Authentication is the most important aspect to distributed system security [20]. To prevent illegal users gaining access to a node or to guard against illegal nodes or service providers gaining access to network resources, authentication is needed. This includes mechanisms to prevent replay or masquerade attacks. Users who need access to multiple computers are inevitably allocated at least one identity on each and then have the problem of remembering a mass of passwords. Therefore users prefer single sign-on systems which permit the user to authenticate themselves once and then have their confirmed identity propagated to the multiplicity of computers that are accessed. It is easy to authenticate users through the following techniques:

- Passwords
- Dynamic passwords using token devices
- Biometric authentication

Authentication measures for network entities can be classified as follows:

1. Challenge/response authentication using secret key cryptosystems:

In this measure, an authentication entity authenticates an entity by sending a challenge and receiving a response. The authenticating entity compares the response with the calculated result using the shared secret key. A three-way communication will authenticate both the sender and receiver. This requires an appropriate algorithm for strong confidentiality to be chosen, because the authentication key may be disclosed by means of attack.

2. Authentication using public key cryptosystems:

An authenticating entity authenticates by receiving the ID or the text encrypted by a sender's secret key and retrieving the ID or text by using the public key. This is similar to authentication by means of a digital signature. The only problem is that the algorithm is complex and authentication takes longer than when using secret key cryptography.

Both systems incorporate originality and integrity mechanisms such as nonces or time-stamps to prevent replay attacks. Systems incorporating time-stamps are most secure although it places stringent requirements on a synchronized timing scheme within the network. By allowing the time tolerance to be too large replays will become possible although a time tolerance that is too short might wrongly reject valid information that was delayed by network latency. In systems that lack synchronized time a three-way authentication using nonces are used [21].

Although method one is secure the scenario of each entity having a shared key is unlikely. Method two is therefore more practical as an entity's public key is easily obtained. The ITU-T X.509 recommendation is a framework where trusted certificates allow for authentication. X.509 is based on the use of public-key cryptography and digital signatures. X.509 does not specify a specific algorithm, but RSA is recommended. The digital signature scheme needs a suitable hash algorithm which will be discussed further in section 2.2.3. This system requires a trusted CA to provide certificates. The certificate is signed by the CA and can therefore be verified by anyone with the CA's public key. No entity except the CA can modify the certificate. The certificate contains the following elements:

- Version
- Serial number
- Signature algorithm identifier
- Issuer's name
- Period of validity
- Subject's name
- Subject's public-key information
- Extensions
- Signature

The ITU-T X.511 further defines tokens that is used to authenticate specific entities and exchange a secret session key once both parties obtain each other's certificates. The token is signed with the issuer's secret key. It contains a timestamp and has a short validity time. This, along with the integrity, prevents the changing of token information. The

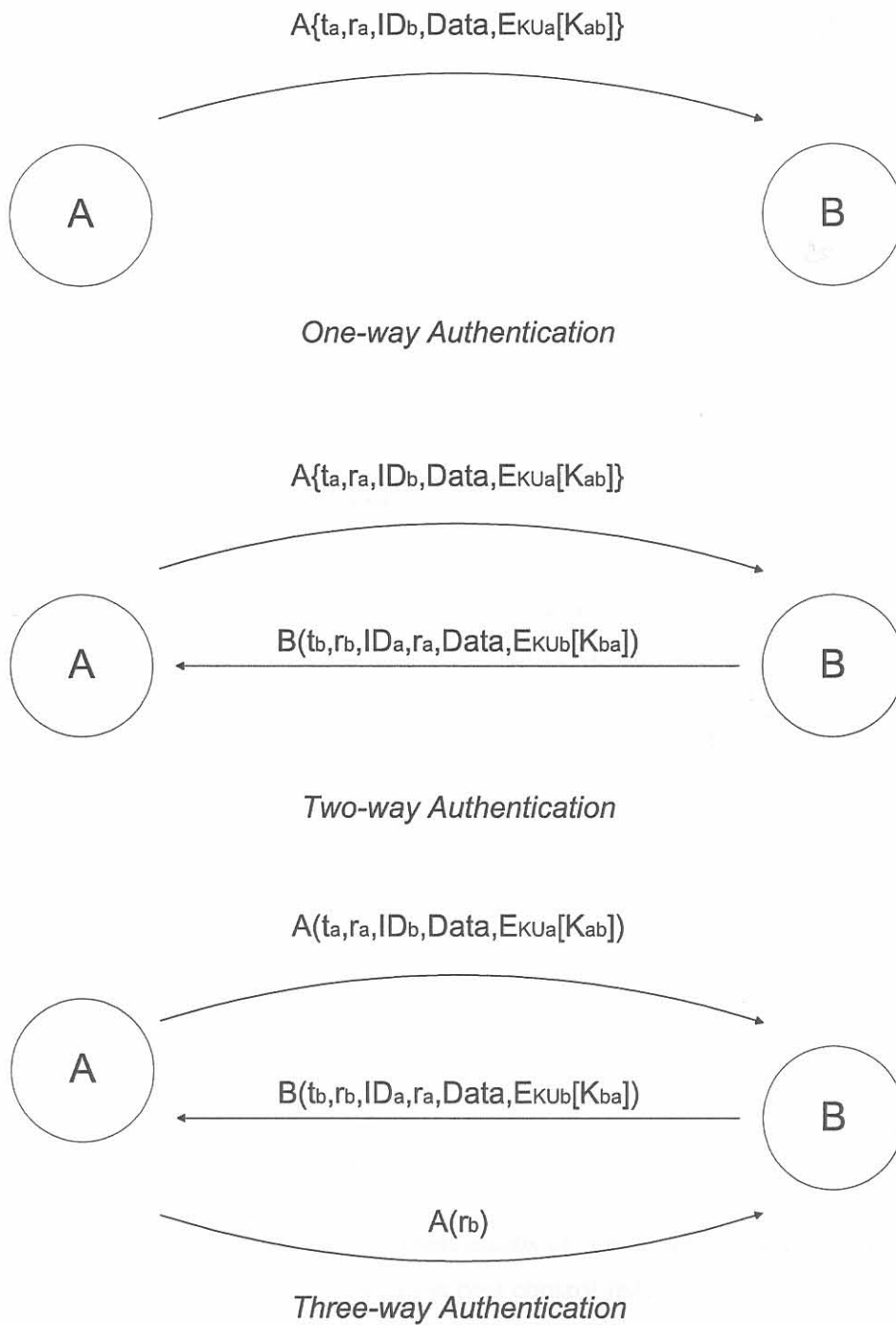


Figure 2.3:
X.509 Strong Authentication Procedures

random number and the timestamp prevent replay of the attack. The target must store all token identifiers within the validity time window in order to detect attempts of replay.

The three authentication schemes proposed by the X.509 standard using X.511 tokens are shown in figure 2.3.

- Message M encrypted by the key K is denoted $E_K[M]$
- KU_x is x 's public key.
- t_x and r_x is a timestamp and nonce from entity x
- ID_x is the identifier of entity x
- Message M signed by x is denoted $x(M)$

2.2.3.2 Access control

Access control is the ability to limit and control the access to host systems and applications via communication links. To achieve this the entity trying to gain access must first be identified, or authenticated, so that access rights can be applied to the individual. It provides protection against unauthorized use of resources and controls access to

- communication resources,
- reading, writing or deletion of an information resource, and
- execution of processing resources.

Logical access control is used to protect a computer and its information. Two sub-processes can be defined [22]:

- The user or remote system wishing access to the resources and data on a local system must be authenticated.
- The logical access control system then allows or denies access to resources based on some predefined criteria such as an access control list.

An entity must be assigned implicit or explicit rights for accessing a service. In other words the entity must be authorized to access services. Identity-based access control involves authorization criteria based on specific individual attributes. It is sometimes referred to as discretionary access control because authorization is performed at the discretion of the service owner. It is usually expressed in the form of an access control matrix.

The rows represent subjects (users, etc.) and the columns represent objects (files, service, etc.). The intersection shows the type of access the subject has to the object. In an information system with many security levels, it is not possible to enforce security with only an identity-based control policy. Discretionary controls regulate the accessing of objects, but do not control what subjects might do with the information therein. For this purpose rule-based access policies are used. These are based on a small number of general attributes or sensitivity classes that are universally enforced. Thus, all objects of the protected system must be marked with security labels. This type of access is referred to as mandatory access controls or information flow control [9].

The node used will inherit the user's security attributes in order to facilitate access control to various services [23]. Each authenticated user will have security attributes similar to these:

- Authentication level
- Group
- Role
- Confidentiality class
- Entity identity

This dissertation will not discuss in detail the rules regarding access control or any effort made to determine whether a certain user has rights to access specific services or information. It is assumed that adequate access control can be implemented if the authentication protocol is sound and secure. This includes features to prevent attacks from the public network. A firewall or access list at the gateway should be set up and resources managed in such a way that only traffic from trusted parties are permitted.

Malicious programs must be taken into consideration and prevented from entering a secure environment. A gateway with sufficient scanning software and user policies for network hosts can prevent these programs from propagating into the system. This is generally not a problem in embedded systems as software are preprogrammed in hardware and therefore malicious code cannot be added afterwards.

2.2.3.3 Confidentiality

This is the protection of transmitted data from attacks involving information leakage. With respect to the content of a data transmission, several levels of protection can be identified.

- Connection confidentiality: The protection of all user data on a connection.
- Connectionless confidentiality: The protection of all user data in a single data block.
- Selective-field confidentiality: The confidentiality of selected fields within the user data on a connection or in a single data block.
- Traffic-flow confidentiality: The protection of the information that might be derived from observation of traffic flows.

Confidentiality ensures that information is not made available or disclosed to unauthorized individuals. The only way to implement this is by cryptographic techniques. This is probably the easiest service to provide taken that implementation is basically a choice of a sufficiently strong algorithm. The choice of algorithm depends on key management, security level and available hardware. Algorithms are divided into two groups [22]:

- Asymmetric: These uses public-key cryptography such as RSA, Diffie-Hellman, Elliptic Curve.
- Symmetric: These uses shared secret keys such as DES, 3-DES, AES (Rijndael), Blowfish.

If a secure key exchange mechanism is available symmetric encryption is the better option. It is faster, in software and hardware, and therefore provides better bulk encipherment. The popular DES algorithm, 64-bit block length with 56-bit key, has been relegated to legacy systems in recent past due to security risks. The AES standard is a symmetric block cipher with a block length of 128 bits and support for key lengths of 128, 192 and 256 bits. This algorithm has clear benefits in speed and strong security and is therefore gaining in popularity.

Another important consideration in confidentiality is the placement of the encryption functions. Two options are available: link encryption and end-to-end encryption. With link encryption, each communication link is equipped with encryption devices on each end. Therefore all traffic over that communication link is secured. This is the most secure

option and also provides unrivalled traffic flow confidentiality. The disadvantage is that, because the encryption is provided in the lowest two levels of the OSI model, the data must be decrypted every time it enters a network device such as a switch or router. All the potential links in a path must incorporate link encryption and each device must share a secret key with each link partner. This involves lots of overhead and also increases the network delay. End-to-end encryption is provided in the upper layers of the OSI model and ensures that the relevant data sent is secure. This scheme requires that only the sender and final recipient share a secret key. This method is faster but offers limited protection against traffic analysis. Traffic padding mechanisms protect against traffic analysis. It is sometime possible for outsiders to draw conclusions based on the presence, absence, amount, or frequency of data exchange. Traffic padding mechanisms keep traffic approximately constant, so no one can gain information by observing it. Traffic padding is achieved by sending encrypted random data over the network.

2.2.3.4 Integrity

Integrity mechanisms provide assurance that data received are exactly as sent by an authorized entity and give protection against threats to validity of data. As with confidentiality, integrity can apply to a stream of messages, a single message, or selected fields within a message. Again, the most useful and straightforward approach is total stream protection. Five types of integrity are defined [14].

- Connection integrity with recovery: Provides for integrity of all user data on a connection within an entire data sequence, with recovery attempted. It detects any modification, insertion, deletion or replay attacks.
- Connection integrity without recovery: As above, but provides only detection without recovery.
- Selective-field connection integrity: Provides for the integrity of selected fields within the user data of a data block transferred over a connection and determines whether the selected fields have been modified, inserted, deleted or replayed.
- Connectionless integrity: Provides for the integrity of a single connectionless data block and may take the form of detection of data modification. Additionally, a limited form of replay detection may be provided.
- Selective field connectionless integrity: Provides for the integrity of selected fields

- within a single connectionless data block and determines whether selected fields of the message have been modified.

The mechanism is used to ensure that information is not modified, inserted or removed by unauthorized individuals, entities or processes. The protection is provided through adding some authentication information to the plaintext before encryption. Another way is to use a digital signature mechanism. Digital signatures provide not only data integrity but also non-repudiation. If only integrity is required a message authentication code should suffice.

A MAC is based on applying a cryptographic hash function, $h()$ to the data that must be protected. Hash functions are much faster than conventional cryptography mechanisms. If a cryptographic hash function is applied to an input value of any length the resulting output value will always be of a constant length. The fixed length output is referred to as the message digest, checksum or hash sum. The MAC is computed in the following way:

$$MAC(M) = f(secretkey, M) = h[secretkey, h(secretkey, M)]$$

If the sender and receiver both know the secret key, the receiver can check the sender authenticity and message integrity by applying the combination of known cryptographic hash functions to the secret key and message. The most popular cryptographic hash function family is the MD family, although MD5 is specified in most documents issued by the IETF and is the latest in the family. Its 128-bit output is potentially vulnerable to birthday attacks and it is believed that it also has structural problems. SHA-1 is a better choice since it produces a 160-bit output. The input message can be up to 2^{64} bits long. The SHA-1 standard describes two methods of computation: One takes longer and uses less memory while the other executes fastest but requires more memory.

The very idea of a digital signature is that the receiver of a digital message should be able to verify the origin and integrity of the message, preferably using only public information. Digital signatures must be message dependant as well as signer dependant. There are two popular digital signature schemes based on public-key cryptography that are described by the DSS: RSA and DSA [24].

RSA requires the following public parameters [14]:

- large primes p and q

- $n = pq$
- $d \equiv e^{-1} \text{ mod } \Phi(n)$
- Private key d
- Public key (e, n)

The RSA encryption is based on the following principles:

$$\text{Ciphertext} = M^e \text{ mod } n$$

$$\text{Plaintext} = C^d \text{ mod } n = (M^e)^d \text{ mod } n = M^{ed} \text{ mod } n$$

Using RSA as the digital signature technique we need to generate S as follows:

$$S = K_{\text{Private}}(h(M))$$

The hash function is of fixed length and is usually rather short compared to the whole message. The process of generating a signature is computationally expensive so it is more efficient to use the hash than the entire message. To verify the signature it is necessary to have M , K_{Public} and the signature S . The verifier can compute the hash and compare it to the decrypted S to verify integrity:

$$\text{Is } K_{\text{Public}}(S) = h(M) ?$$

Until recently some countries prohibited the use of an algorithm in signatures that could also be used for encryption. This is the reason DSA was originally developed. DSA is based on the discrete logarithm problem and is related to the ElGamal algorithm [14]. It required much more computation than RSA in order to verify. DSA requires the following public parameters:

- large prime p
- large prime q , $q | (p - 1)$
- generator modulo p of order q , g

The signer keys consists of:

- random integer x
- $y = g^x \text{ mod } p$

The signature consists of a number pair (r, s) computed in the following way:

$$r = (g^k \bmod p) \bmod q$$

$$s = [k^{-1}(h(M) + xr)] \bmod q$$

To verify the signature the verifier computes:

$$w = s^{-1} \bmod p$$

$$u_1 = h(M)w \bmod q$$

$$u_2 = rw \bmod q$$

$$v = (g^{u_1}g^{u_2} \bmod p) \bmod q$$

Signature is valid if $v = r$

Detection of duplication replay and loss of messages is achieved with sequence numbers/time stamping before hashing. This gives the recipient the ability to verify that exchanges genuinely took place during the time interval that the time-stamp defines.

In a closed system with trusted entities integrity could be implemented using a message authentication code (MAC). This however is difficult if every node does not share keys with another and it does not provide non-repudiation. Therefore the integrity mechanism generally used in networks is built from hash functions using the RSA digital signature method.

2.2.3.5 Non-repudiation

Non-repudiation prevents either sender or receiver from denying a transmitted message.

- Non-repudiation, Origin: Proof that the message was sent by the specified party.
- Non-repudiation, Destination: Proof that the message was received by the specified party.

According to the general framework for non-repudiation as defined by ITU-T Recommendation X.813 the service comprise the following:

- Generation of proof
- Recording of proof

- Verification of proof generated
- Retrieval and re-verification of the proof

The technical means used to ensure non-repudiation services are the electronic signature of documents, the intervention of a third party witness and time stamping. Non-repudiation is also a legal concept and must also be defined by law. For public key cryptography each user is the sole owner of a secret key. Unless the whole system is compromised a given user cannot repudiate a message accompanied by his/her signature. Symmetric key mechanisms require a trusted third party witness as two entities possess the same key.

2.2.3.6 Availability

Availability can be defined as ensuring that a system's services are accessible on demand by authorized users. This service addresses the security concerns raised by denial-of-service attacks. It depends on access control and proper management and control of system resources.

Anonymous interference with communication such as denial of service must be addressed. Although there is not a specific security mechanism to provide availability it must still be considered when implementing a security policy. Sufficient authentication and access control mechanisms should prevent disruptive entities from gaining access to resources. Even then resources should be managed in such a way that no single entity can engage any resources in such a way that another entity's usage of that resource is restricted. This can be done by ensuring that entities only accept authenticated connections.

2.2.4 Security Assurance

To ensure the secure operation of a security policy infrastructure, it is necessary to have some accepted practice for the identification of security risks as well as the application of appropriate controls to manage risks. This practice is simplified by the use of formal methods and tools which increase the reliability of the system specification. Some of the relevant guideline are [25]:

- ISO/IEC 13335 or GMITS provides guidance on the management aspects of IT security.

- ISO 17799/BS7799 is a code of practice. It offers guidelines and voluntary directions for information security management. It is meant to provide a high level, general description of the areas currently considered important when initiating, implementing or maintaining information security in an organization.
- ITSEC is the existing European IT security evaluation criterion standard that allows security certification to be granted from qualifying certification bodies. ITSEC was the first example of formal security recognition between nations [26].
- COBIT provides good practices for the management of IT processes in a manageable and logical structure, meeting the multiple needs of enterprise management by bridging the gaps between business risks, technical issues, control needs and performance measurement requirements.
- Common Criteria represents the outcome of international efforts to align and develop the existing European and North American criteria towards a common standard for carrying out security evaluations [27]. By establishing a common base, the results of an IT security evaluation are more meaningful for a wider audience. CC has a catalogue of standard security functional requirements that represent the current state of the art for trusted products and systems. These can be used to develop a protection profile and as a means for developing a security target. They can also be supplemented or tailored to suit more specialist requirements. A CC evaluation is carried out against a set of predefined assurance levels, termed Evaluation Assurance Levels (EAL0 to EAL7). This scale represents ascending levels of confidence that can be placed in the TOE's security functions and determines the rigour of the evaluation:
 1. EAL1: Functionally tested
 2. EAL2: Structurally tested
 3. EAL3: Methodically tested and checked
 4. EAL4: Methodically designed, tested, and reviewed
 5. EAL5: Semi-formally designed and tested
 6. EAL6: Semi-formally verified, designed and tested

These standards provide guidelines to follow when developing a security policy. They do not provide specific solutions or suggestions on policy implementation. They rather describe a risk management methodology which ensures that a security policy is formulated

system also allows for easier authentication, integrity and non-repudiation mechanisms. The gateway is assumed to be trusted as it can easily be regulated that only one gateway is allowed per private network. The gateway is therefore a viable point of distribution for X.509 certificates on the private network. On the public network a trusted CA such as Thawte [28] or Verisign can be utilized. The X.509 standard also specifies the steps for creation, revocation and destruction of keys.

The risk that a key is compromised increases with time and usage. Keys have to be replaced regularly without causing service interruption. The user loses the right to a private key if the key is revealed or the secret key of the CA is compromised. All associated certificates must be revoked and the revocations communicated to all relevant verifying entities.

2.3 DISTRIBUTED FIELD AREA NETWORKS

DMC (Distributed Measurement and Control) systems are widely used in industry today. Older centralised models had one intelligent point that relayed commands to dumb nodes. In the distributed system the nodes have intelligent capabilities that allow them to function with little guidance from external sources. This makes the system more reliable and allows for more efficient use of resources. A model for a distributed system is shown in figure 2.5.

The enterprise level consists of the end user who has the ability to adjust specifications of the system. The user can also request information regarding the status of the process or the nodes. The application layer contains the protocols for the applications that is run by the nodes and relays data between the user and the node. The distributed intelligence level contains the intelligent datanodes. These nodes gain adequate knowledge from other nodes or the application layer to measure and control their respective processes. The process connection level is the way the node interacts with the process either by means of actuators or sensors. DMC systems should preferably be designed using standardised protocols that will reduce the cost of the product and promote interoperability. The nodes should have a standard interface and allow for easy maintenance and 'plug-and-play' sensors and actuators. All these aspects makes a viable DMC system with a long lifetime and maximum functionality [29]. Due to hardware considerations and the reliability required from communication networks in robust industrial systems a number of so-called fieldbus

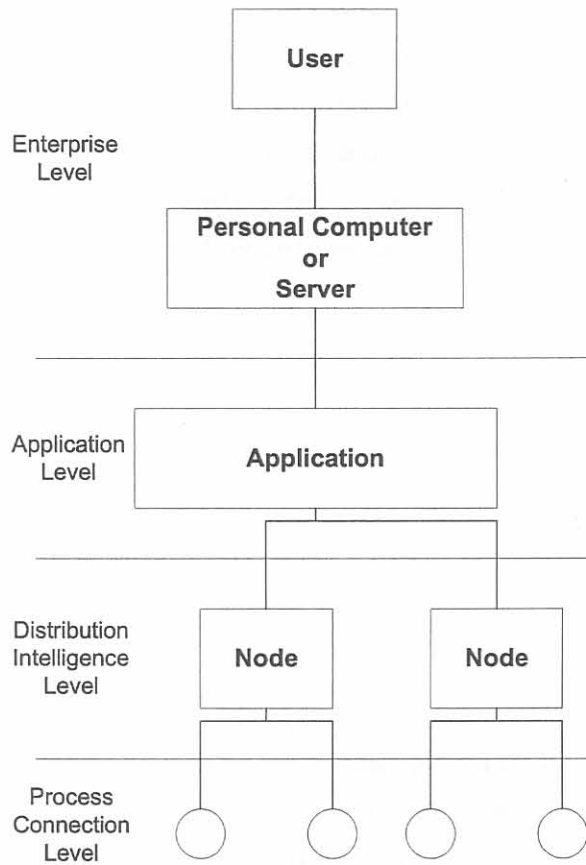


Figure 2.5:
Distributed system model [30]

protocols have been developed, such as LonWorks, Profibus, CANBus and BACNet [31]. Although not offering the connection-oriented, high-bandwidth functionality of TCP/IP Ethernet these protocols are easier to implement, offer better error-correction and are regarded as robust and dependable, which are desirable qualities in embedded industrial systems.

The low cost of computer equipment has led to a change in the way network services are delivered to end users. In recent times the computer network industry has moved away from the traditional server-client model. Information and services are no longer restricted to secure centralized control rooms and autonomous network infrastructure. Applications and information are increasingly distributed across a variety of platforms involving a mixture of vendors, technologies and security measures. This distributed network system brought many advantages [32]:

- Reliability - Accessible alternatives reduces service failure.
- Sharing of resources - Enables anyone to share or trade with everyone else.
- Aggregate computing power - Instead of one 'supercomputer' running everything the responsibilities are shared.
- Scalability - Distributed networks adapt easily to any possible number of users or services.

Real-time networks generally have time-constraints placed on its performance. Each unit of work that is to be performed is called a job and a set of related jobs that jointly provide a service is called a task. It is common to divide timing constraints into two types: hard and soft. According to commonly used definitions a timing constraint is hard if the failure to meet it is considered a fatal fault. A hard deadline is imposed on a job because a late result produced after the deadline may have disastrous consequences. In contrast the late completion of a soft deadline is undesirable but will not have a serious consequence. The requirement that all hard timing constraints must be validated invariably places many restrictions on the design and implementation of hard real-time applications as well as architectures of hardware and system software to support them. The developer of a soft real-time system is rarely required to prove rigorously that the system meets timing objectives. This allows the developer to concentrate on other performance features as well.

2.3.1 Security in Embedded Systems

Embedded systems are quickly becoming ubiquitous within our daily lives. Most people interact with an embedded system at some point during their day without knowing it. The embedded system may be in their automobile, refrigerator, or cellular phone and range from personal digital assistants to disk controllers and home thermostats to microwave regulators [4]. The key trend, however, is that all such devices are becoming more powerful, autonomous, and highly connected - following essentially the same growth curve as the Internet. The one major area in which security is addressed with regard to embedded networks is home automation. Long the futuristic domain of hobbyists, home automation is moving into the mainstream. Utility service providers offer specialized and valuable applications such as energy load management to retain customers in the face of competition. More new homes are wired for intelligent control, with security, comfort, and convenience systems becoming network aware. Homeowners can orchestrate and monitor appliances from multiple locations within the house or even remotely via telephone and the Internet, and delegate limited controls to utility service providers. By virtue of their ubiquity these systems must remain transparent to users if they are to be successful [8].

The dynamics and market forces are similar, we might suppose that the problems with security and privacy are similar as well. It would be wise to learn from past successes and failures and apply these lessons within embedded systems. Given the size and computational requirements of popular security protocols (SSL and SSH) and encryption algorithms (RSA and triple-DES), how to accomplish this task is unclear. To implement security in embedded systems is dramatically different than for full-featured, general-purpose computers. Even with today's advanced technology, embedded systems typically have severely limited resources [33]:

- Volatile and non-volatile storage are usually orders of magnitude smaller than in general-purpose computers and embedded CPU speed and available bandwidth are orders of magnitude slower.
- The capabilities of embedded systems are approximately 10 to 15 years behind the general-purpose market. Yet we still expect these systems to provide today's security levels - not those of a decade ago.

Efforts such as the Advanced Encryption Standard (AES) from the National Institute of Standards move in the right direction as the requirements for the AES algorithm included

several issues helpful to embedded systems. However, AES does not solve the entire set of problems. As a result many developers implement proprietary protocols and algorithms. This is a dangerous proposition because the approach lacks significant peer review. While it can be done, doing it right is difficult. Therefore it is required that accepted and certified technology be used. A great deal of research is still needed to provide robust security protocols supporting embedded systems [34].

Many people incorrectly view security in isolation. A single security mechanism or a single certification cannot provide adequate security. Instead, we must view security holistically, taking the overall composition of the security mechanisms and processes into consideration. This is what makes providing security an extremely difficult task [35].

It must also be accepted that no embedded hardware will ever be completely secure. IBM's 4758 is a physically secure co-processor for protecting both data and computation in potentially hostile environments. In addition to providing physical protection, its design goals encompassed the equally challenging problems of securely downloading applications into the secure environment and remotely identifying and authenticating the embedded device. The IBM 4758 was the first device to obtain a FIBS 140-1 Level 4 validation, the highest level of commercial cryptographic certification currently available. The IBM 4758 secure co-processor however has some protocol flaws. These faults make it possible to extract application secrets by following an avenue of attacks that exploit fundamental design flaws in the mathematical properties of protocol operations instead of the protocol implementation flaws that code-injection attacks exploit (e.g. buffer overflows)[25].

2.4 SMART CARD TECHNOLOGY

Smart cards in general are tamper-resistant computer microprocessor chips. They have the ability to run applications to make computations on data using programs stored in memory. This section discusses hardware architecture, software, benefits and possible applications of smart card technology.

2.4.1 Smart Card Architecture

The physical support for a smart card is a plastic rectangle on which information about the application or the issuer as well as readable information about the bearer, for example

name, date of validity, photograph, can be printed. The support can also carry either an extra magnetic strip or a bare code label. An array of eight contacts is located in accordance with an international standard. Six of these contacts are linked to the chip, which is usually not visible. They are used for power supply, ground, clock, reset and a serial data communication link [36].

A typical architecture for a smart card comprises five main components:

- The processor. This is often an 8 bit processor, the most common being Motorola's 6805 and Intel's 8048. New devices are beginning to appear in a few recent and powerful cards.
- A working memory. This is used to store temporary data when the card is in use. It is also known as the RAM.
- A ROM program memory. This contains permanent code to be executed by the processor. It should be noted that this program is stored through a mask and cannot be changed in any way.
- Non-volatile data memory. This type of memory can be written and erased over thousands of cycles.
- A communication device for exchanging data and control information between the card and the external access terminal. This communication unit works in the same way as any serial asynchronous link. The most frequent bit rate is 9600 bit/s.

For the purpose of better performance, there is often a separate cryptographic coprocessor (e.g. a modular arithmetic coprocessor for public key computations). The input/output parts and the power source differ for different types of smart cards: there are contact cards with metallic contacts, contactless cards using inductive coupling, and super smart cards with a keyboard and a display. A processor chip of a typical smart card contains three different types of memories: the working memory RAM, the maskable memory ROM, and the data storage EEPROM. The procedures and, if possible, cryptographic algorithms for general use are stored in the ROM. When an application running on an application terminal (e.g. a PC) wishes to communicate with a smart card, the card must be inserted into a card reader (also called card terminal or card accepting device).

The most important international smart card standard is the ISO/IEC 7816 [37]. This standard ensures physical compatibility between integrated circuit cards with contacts and card readers. The first specifications have focused quite naturally on the physical dimensions of the card, position of the contacts, power supply, share and duration of the electric signals, and protocols for communication between the card and the terminal. With the increase in commercial and telecommunications applications of smart cards, other parts have been added. Today's standard consists of six parts, with more to come:

- ISO 7816-1, the oldest part, specifies the physical characteristics of the card, the dimensions of the integrated circuit, the resistance to static electricity and electromagnetic radiation, the flexibility of the support, and location of the integrated circuit on the card.
- ISO 7816-2 defines the dimension and the position of the metallic contacts on the card.
- ISO 7816-3 describes the electric signals (polarity, voltage, duration, etc.), transmission protocols between the card and terminal, and the card's response to a reset originating from the terminal. Four protocols are currently defined:
 - A character-oriented half-duplex protocol identified by the value $T=0$.
 - A block-oriented half-duplex protocol identified by the value $T=1$.
 - A block-oriented full-duplex protocol identified by the value $T=2$, although this mode is rarely used.
 - The value $T=14$ indicates the use of proprietary protocols, used to support applications that preceded the standard and were already planned in France and in Germany in the health field [38].
 - The values $T=3$ to $T=13$ are reserved for future use.
- ISO 7816-4 defines the local organization of the data stored in the card and the framework for secure access to these data, in particular:
 - Cardholder authentication using a password (the PIN).
 - Authentication of an external entity using a secret key that authenticates the terminal or the bank.
 - Verification of the data integrity using a cryptogram that is often a message authentication code.

- Encryption of the data.

The ISO 7816-4 commands fall in three categories: administrative commands, security commands, and communication management commands. In general, card manufacturers prefer to pick and choose from the list of commands, so most of the cards commercially available provide only a subset of the ISO 7816-4 commands. Additional proprietary commands will be added to facilitate file and data management.

- ISO 7816-5 defines the procedure to register the application to obtain a worldwide application identifier (AID).
- ISO 7816-6 defines the inter-industry data elements.

From a technical point of view, smart cards can be classified into three main families, automata, microprocessors with simple data management and microprocessors with high-level data management. This last family is often known as multi-application cards. A common characteristic of all three families is a potentially high level of security comprising encryption using various algorithms and a distributed system of secret keys. This has been made possible by the capability of executing the required algorithms within the card itself. Such a capability does not exist with passive optical and magnetic cards [39].

The first rule of security is to gather all these five elements of a card onto a single chip. If this is not done the external wires, linking one chip to another, could represent a possible route for illegal access or use of the card. ISO standards specify the ability of a card to withstand a given set of mechanical stresses. The size of the chip is consequently limited and most of the actual constraints follow from this limitation, especially the data memory size. Chips for cards are very reliable and most manufacturers guarantee the electrical properties of their chips for ten years or more. ISO standards specify how a card must be protected against mechanical, electrical or chemical aggressions. For most existing applications a card is obsolete before it becomes damaged.

Smart card security issues can be divided into four areas:

- Card-body security.
- Hardware security.

- Operating system security.
- Card application system.

Most card-body security measures, such as embossing or hologram pictures, are designed to allow humans to check whether a card is genuine.

The smart card microcontroller (i.e. chip) must be as tamper resistant as possible. This effectively means that the cost of breaking the chip security mechanisms must be higher than the potential gain from doing so. It should be impossible to read the secret data stored on the card, such as cryptographic keys, or monitor processes running on the card and thus draw conclusions about sensitive information. Attacks against chip security can be performed at any phase of the card life cycle - card development, card manufacturing, card personalization or card use. Different attacks are performed when the chip is active (i.e. has a power supply) or inactive. Therefore it should be noted that tamper resistance does not solve all security problems and must be carefully analysed and upgraded if necessary. Security measures during card development and manufacturing include control of physical access to card data. It is also very important to implement only documented features, because undocumented features are not considered in evaluation and testing and thus can open a security hole. Each chip obtains a unique serial number, which in itself cannot protect against attacks, but serves as information for deriving cryptographic keys. During manufacture, chips are protected by authorization mechanisms based on transport codes, which can even be chip specific [36].

Most attacks on smart card hardware are performed during card use because there is practically no physical access protection. For such attacks, various rather sophisticated tools may be used, such as microscopes, laser cutters, micromanipulators, or even fast computers for probing and analysing the electrical processes on the chip. Static analysis can be made extremely difficult through special design principles such as:

- Embedded of tamper-detection mechanisms such as cover switches or motion detectors to detect, for example, cutting or drilling;
- Opaque tamper-evidence coating to hamper direct observation, probing, or manipulation of the chip surface;
- Dummy structures to confuse attackers;

- Special memory design and scrambling to hide content;
- Hiding and scrambling of buses to prevent eavesdropping.

Mechanisms that protect against dynamic analysis include:

- A voltage watchdog that switches off a chip module if the power voltage is not within a specified interval;
- Mechanisms that set to zero any parameters representing secret or private information (i.e. cryptographic keys);
- Environmental failure protection that shuts down the chip or sets sensitive parameters to zero whenever environmental conditions are outside the normal operating range (i.e. chip heating).

A dynamic attack that can determine which card command is being executed on the card (and thus potentially reveals sensitive information) is based on differential power analysis [40]. The attack works if different commands have different power consumption, so one protection mechanism is to use only commands with very similar power consumption. Another possibility is to perform the same computation (e.g. in a cryptographic algorithm) in several different ways, so that each time one way is chosen randomly. Another well-known attack is the timing attack, in which time intervals needed by the card for specific computations are measured and analysed. For example, if the card encrypts data, the greater the differences in the duration of computation for different keys and data, the easier it is to reduce the set of possible keys. A protection mechanism is to make the duration of cryptographic computations independent from input data. Attacks based on differential fault analysis try to disturb the functioning of the card (e.g. by changing the power voltage or the frequency of the external clock, or by exposing the card to different kinds of radiation). Each time the card performs symmetric or asymmetric cryptographic computations, one bit in the key is changed at some position. The result of a series of such computations, which are all different because the bit position is different in each, are analysed and used to compute the previously unknown key. The simplest protection mechanism is to let the card perform each cryptographic computation twice and to compare the results. This method is, however, rather time-consuming. A more practical approach is always to append a random number to the data to be encrypted so that attackers cannot analyse different results for the same plaintext. The random

number generator on the smart card should ideally never repeat the random numbers at any time during the card life cycle.

At present IC cards work as slaves. The program that is contained in the ROM is only an interpreter of commands coming from the outside. The protocol between the card and the co-operating device is partly standardised. It begins with a 'Reset' command that is sent to the card by the device. The response of the card is used to identify the card with respect to manufacturer information (e.g. manufacturer's code, type of card, serial production number, baud rate, type of protocol), application information (e.g. application code, security scheme) and eventually owner information (e.g. status of the card, personal security code).

There are four types of commands. The first type is used to organise the logical storage and the security scheme. For example: create or delete a logical area, give this area a name and a size, create a protection for this area or link and store a secret code. These commands are usually only available to the issuer of the card during the first session, which is called the personalisation step. The second group of commands is used by any application to manage the security scheme, i.e. to verify that any physical or logical partner that takes control of the card is actually authorised to access some of the existing areas. For example: present a secret code, present a personal identification number. The third group comprises input and output commands or, more generally, data manipulation commands: read, write, update, increase (counters for token-controlled services), decrease, compare, search, etc. The fourth group is used to add extra functions such as encryption, random-number generation, requesting unused memory size and unlocking a locked card (if that card locked itself because it has detected some attempt to bypass the security scheme). At present some vendors include some commands for non-standardised features but most of the commands ensuring basic functionality are standardized.

As a consequence, three different elements of software can be observed in a card application. The card manufacturer supplies the internal code. It is written in the enclosed microprocessor machine language. All cards of a given application contain data that are used to describe both the security scheme and the data structure. This information provides a common basis for allowing an external device to exploit all cards of the application in the same way. Every card contains specific data that characterises the owner. Finally, the external world, comprising a reader, a PC or a mainframe computer accessed through

a network, executes a set of programs to communicate with the card and manage the data held in it.

The file system defined in the standards supports two categories of files, dedicated files and elementary files. Each file has an identifier coded on 2 octets in hexadecimal notation. Figure 2.6 illustrates the way the files are arranged.

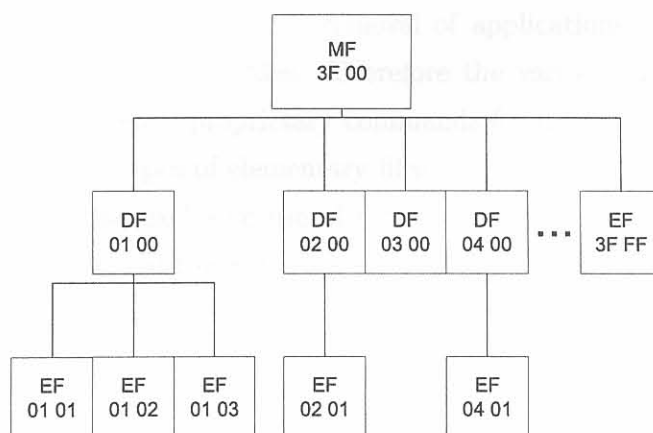


Figure 2.6:
Three structures of ISO 7816-4 file system [21]

The master file (MF) is at the root of the tree and is always identified by the file identifier 3F 00. The file identifier of the first DF is 01 00 and the last is 3E 00. Thus a card cannot contain more than 62 dedicated files in addition to the master file. Each DF is associated with a given application and may contain one or more elementary files. Application selection may be through the SELECT FILE command with the application identifier (AID) as an argument or indirectly with the help of the special elementary DIR (Directory) or ATR (Answer to Reset).

The EFs contain the data. Each EF is identified by its position in the tree, i.e., by the path leading back to the master file. The identifier is also coded on 2 octets and takes the form xx yy, where xx is the identifier of the DF to which the EF belongs. xx is 3F if the file depends directly on the master file. yy is a sequential number of the EF in that particular directory. Thus, the number of elementary files in a directory cannot exceed

63. The elementary files 2F 00 and 2F 01 under the master file have special indexing functions. The first is called DIR and the second ATR. The file DIR contain elements that allow the identification of the applications while the file ATR specifies how the card can find the application or the various objects.

The maximum number of elementary files in a card is thus $63^2 = 3969$ files. This structures is rigid and does not suit dynamic situations where files can be added or deleted corresponding to the addition or removal of applications. In fact, ISO 7816-4 does not allow the creation of new files. Therefore the various suppliers of integrated circuit cards have had to define proprietary commands for file management. ISO 7816-4 distinguishes between two types of elementary files: internal EFs and working EFs. The latter contains data for the exclusive use of entities external to the card. The internal EF contains data that the card uses during its operation. For example, in a monetary application, the following files can be present [41]:

- Key files for the storage of keys that will be used to derive a session key as specified by the payment protocol employed. Given the sensitivity of banking transactions, the applications that use purses will most probably need several keys, one for each action, such as for certification, for debit, for credit, or for electronic signature. Each key will be associated with an individual file.
- PIN files to stock the PINs that control access to the application file. The application files and the access conditions are irrevocably defined during the personalization phase.
- Purse files. For each purse, the file indicates the maximum balance, the maximum payment for each transaction, the current balance, and a backup balance to recover the previous value in case of a failure.
- Certificate files, in the case of public key encryption.
- Application usage files.

2.4.2 Smart Card Operating Systems

Development of card operating systems (COS) began in the early 1980s and today there are a dozen operating systems on the market (e.g. CardOS by Siemens, Cyberflex by Schlumberger, Multos by Maosco). COS must be kept as small (e.g. 16K) and simple as

possible in order to make testing and evaluation easy as well as to make it possible to verify whether the high security requirements are satisfied. The operating system code is written in ROM, which means that once a ROM mask has been defined and possibly millions of cards produced, no changes can be made without considerable loss of image and money. There is a range of mechanisms to make a smart card operating system as secure as possible:

- Performance of hardware, software, and memory test based on checksums at initialisation;
- Operating system design with a modular or layered structure so that error propagation is minimal;
- Hardware support to strictly separate memory regions belonging to different applications (e.g. through the addition of a memory management unit (MMU));
- Access control based on PINs.

A well-known attack is a sudden interruption of power supply, such as when a card is removed from a card reader. If performed at a precise moment, this type of attack may cause serious problems. An electronic purse may be loaded at a terminal and then removed from the reader at the very moment when the balance on the card has been increased. If the card has not yet responded to the terminal or no new audit record has been generated on the card, the terminal will believe that the load transaction was unsuccessful. The best protection against such attacks is always to use atomic transactions. This effectively means that a transaction is performed either completely or not at all. Files access control in most COS' is command based. This means that a specific command must be successfully executed before access is granted. For example, write access may be granted only after the PIN has been successfully verified by a specific command (i.e. VERIFY). An alternative is state-based access control. Basically, a state automaton is defined which specifies all allowed execution flows (i.e. command sequences) on the card. The third possibility is object-oriented access control, in which the object to be protected carries its own access control information.

There are three multi-application operating system smart card platforms (Java Card, MULTOS and Windows for Smart Card) available on the market. Commercial vendors back different technologies: VISA backs Java Card whereas arch-rival in finance, MasterCard, backs MULTOS. In the world of networked computing, Microsoft backs Windows

for Smart Card (WfSC), whereas Sun Microsystems backs Java Card. These rivals are unlikely to co-operate in developing unified standards. There are unique differences between the three MOAS platforms, each providing its own advantages and disadvantages, and each one must be considered if a new smart card system is being developed [42].

MOAS smart cards (MASCs) also need the ability to load card applications that sit side by side and provide the card holder with a variety of separate functions, such as digital identity, electronic cash, medical records and mobile phone subscriber identity module (SIM). The smart card security should be independent of the applications that are loaded to the card. MASCs are at least 20 times more expensive than an old-fashioned plastic card with only a magnetic stripe for storage. Given the cost of these highly secure devices, it makes economic sense to maximise their life span by loading and deleting card applications to meet the issuer's needs after the cards have been issued and without having to recall the cards. Therefore a secure way of dynamically controlling the card content over open networks is required. The way that MASCs are structured is as shown in figure

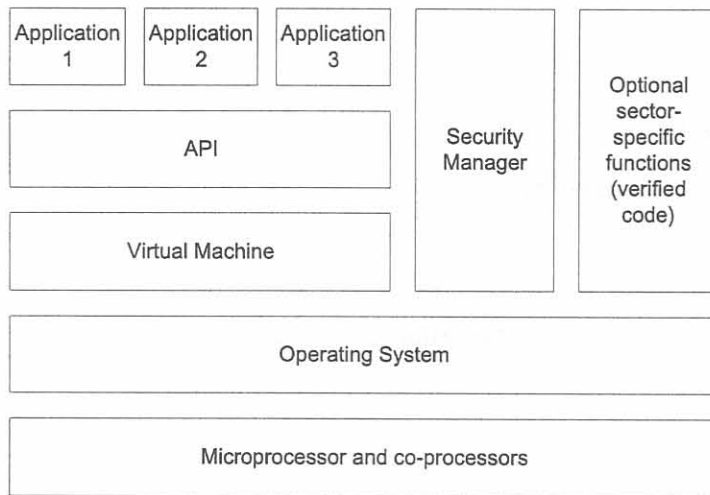


Figure 2.7:
MASC structure

2.7. The basis is a microprocessor chip implemented on a silicon chip with a certain amount of spare ROM and EEPROM. ROM masks are used during the final stages of chip fabrication to add functionality as follows:

- An operating system, allowing access in a secure and controlled manner to the raw processing power of the chip as well as useful libraries for functionality such as cryptography.
- A virtual machine that allows loaded applications to be interpreted and executed is built on top of the OS. These applications conform to the virtual machine API and are perhaps compiled from common high-level languages such as C, Java, Visual Basic, etc.
- A Card manager controlling the security of the MASC, including such functions as loading and deleting card applications and dispatching incoming commands from the terminal to the appropriate application.

It is important to an issuer to obtain supplies of devices from multiple sources and that these devices are interoperable. Therefore the adoption of unambiguous standards is paramount. A brief description of each of the standard MASC platforms that are currently being marketed is given in the following sections. Further details of how the platforms compare are given in table 2.4.2.

2.4.2.1 Java Card overview

Java Card was launched in 1995-96, the first major release was in 1997, and the current release is Java Card 2.1.1 [43]. The specifications are in three parts:

- API: the programmer interface.
- Virtual machine: providing binary portability between Java Cards.
- Runtime environment.

The Java Card Forum [44] develops and recommends specifications to Sun Micro-systems Inc., owner of the Java Card VM and API specifications. Java Cards have had their most success in the mobile telecommunications sector where standards are emerging for SIMs and SIM toolkits. The Java Card standard does not specify an operating system. It also does not specify how Java Cards should dynamically manage their applications. In this

Table 2.4.2:
Summary of MASC products

	Java Card	MULTOS	WfSC	Open Platform
Specification Control	Sun Microsystems	MAOSCO	Microsoft	Global Platform
Current Versions	Java Card 2.1.1	MULTOS 4	WfSC1 .1	OP 2.0
Certification	Unproven	ITSEC E6 - Highest level possible	Unproven	Not applicable
Scope	Java API access run-time environment through virtual machine	Complete OS: Virtual Machine, API, run-time environment, management	Multi-functional, cannot really support many applications	Application management
Portability	Virtual machine byte-code portable, unproven	Proven portability	Not yet	Unproven
Personalization	Personalization after loading application	After or before loading application	Non yet	Only provides life-cycle management
CA Structure	Cost not known	Cost: US\$ 0.04 per application reload	Cost not known	Not applicable
Benefits	Popular Java development tools	Interoperable, secure, stable and efficient	Flexible, familiar Microsoft tools	Global PIN and secure communication channels
Disadvantages	Security, stability, interoperability not proven	Inflexible, no cheaper non-RSA option	Security concerns	Not proven. Bugs in cards and compilers

respect, most Java Cards are migrating towards using the emerging Open Platform standard. Some Java Cards, claiming to be JC 2.1.1 compliant and OP 2.0 compliant, are beginning to emerge. They have yet to demonstrate true interoperability where applications can be compiled once and then loaded onto several Java Card implementations and be guaranteed to execute correctly. Java Card interoperability is further hampered

by the fact that no independent security evaluation is required. All major smart card manufacturers have Java Card platform offerings. Some Java Card licensees claim that they will be targeting Common Criteria EAL4+ and 5+, but it will be the best part of a year before this might be achieved. Java Card applications are developed in Java using standard tools that developers may well already have. However, it is not yet possible to buy a source-level debugger for Java Card applets.

2.4.2.2 Open platform overview

VISA developed the VISA Open Platform [45] specifications to try to fill some of the interoperability gaps in the Java Card specifications. Actually the VOP specifications go further than this in that they also address terminal and personalisation issues. In October 1999, VISA handed over the Open Platform specifications to the Global Platform consortium [46] to manage and encourage the adoption of OP outside financial circles. The OP specifications address the issue of how to manage the card applications. A scheme is proposed that addresses areas such as:

- Dynamic secure loading and deletion of applications
- Secure communication channels between the issuer and the card or card applications
- Global card PIN which can be shared by applications

As with earlier Java Card specifications, emerging platforms using VOP or OP may suffer from incomplete specification and not be able to interoperate. For example, initialisation of the global PIN is specified by VOP but not by OP. The area of most concern is that the control of secure dynamic application management is not specified in complete detail. Implementations of OP will inevitably be incompatible, which will be a disadvantage as far as card issuers are concerned. Open Platform is not limited to Java Cards. In the future we may see OP cards with underlying WfSC technology. In the long term, this may allow some degree of card management interoperability between platforms using OP.

2.4.2.3 Windows for smart card overview

Microsoft's core business is the sale of operating systems. As such they will not allow anyone else to implement Windows for Smart Card operating systems [47]. Microsoft wants to control new technologies (mobile phones, digital TV, palm devices and access to networks) by ensuring the ubiquity of their operating systems. They realise that the smart card will be key to all of these. WfSC has some interesting properties that set

it apart from the other two open standard MASC platforms. One of these is that the issuer can configure the card operating systems and decide which parts should be present on their cards. The intention is that optional modules will include functionality such as GSM and cryptography support, and support for industry standards such as ISO7816-4 [37]. The application developers decide which of these are needed for their applications and the cards can be manufactured accordingly. This leads to the possibility of extremely cheap low-functionality, multiple sources of supply and third-party security evaluation. The \$2 cards that Microsoft promised have yet to appear and are unlikely to if they are to have a useful amount of application memory on them. WfSC does not impose an issuer-centric security model. If the system requires a card where the cardholder controls what applications are on this card, WfSC is the most likely platform. This is much more like the PC model, but has the same security implications.

Since the announcement of WfSC in 1998, Microsoft has repeatedly failed to deliver on its promises. While WfSC has come a long way from the first developer release in May 1999, it is still to be proven to work reliably and be mature. The current version WfSC v1.1 is more of a multi-function smart card than a true multi-application smart card. It does not fully conform to the industry standard for communicating with smart cards at the application level, ISO 78164. While WfSC is not yet a mature commercial product, what is certain is that, with Microsoft's resources, they will have a credible offering eventually. The power behind the name of Microsoft has already proven enough to make large international organisations align with WfSC. Microsoft is also a tool provider. By making familiar tools available at give-away prices, they are guaranteeing that many developers will experiment with creating applications for their MASC. Only the Microsoft Visual Basic compiler is available at present, which produces highly inefficient code, though better ones are likely to appear as WfSC becomes mature. In common with Java Card, WfSC does not specify how the card content is managed securely post issuance. Microsoft spokesmen say that they will wait to see what the market requires. Open Platform is likely to be an option.

2.4.2.4 MULTOS overview

MULTOS has merged from the banking sector [48]. It was developed by the NatWest Development Team in the UK as a secure platform for an electronic purse. It is the only non-military product in the world to achieve ITSEC E6 High certification, which is the highest certification available [26]. ITSEC is the European predecessor of the emerging Common Criteria for Information Security Evaluation. A consortium called MAOSCO controls the MULTOS specifications. As MOASCO believes that security is important, it insists that any implementation of MULTOS is evaluated and achieves certification to ITSEC E6 (High). This is done at the cost of the implementer and takes typically over a year to achieve.

MULTOS is a genuinely interoperable unambiguous standard. MULTOS has included everything required of a MASC (the OS, virtual machine and card manager). If you buy a MULTOS card you know exactly how it will work and switching between suppliers is done with minimal effort. The MULTOS specifications are mature and have been stable for around three years. The MULTOS virtual machine (or Application Abstract Machine, as they call it) is tailored for smart cards and as such allows very efficient (in both speed and size) applications to be written. In order to develop for MULTOS, compilers for C or Java are required, or machine-code-like native MEL may be used for ultimate efficiency (though this is rarely necessary). Only one good compiler toolset is available for MULTOS. It is not possible to ask for a MULTOS card without the secure dynamic application management facility (which uses RSA cryptography), or without the ability to perform cryptography for digital IDs. Thus MULTOS requires a co-processor for RSA and therefore uses relatively expensive underlying chips. However, most smart card applications anyway require RSA (digital ID, Windows 2000 logon, EMV, CEPS e-purses). The secure application management uses digital certificates, which at present can only be obtained from the MULTOS CA based in the UK. This model may be unattractive to potential issuers and MAOSCO plans to offer to license the CA to third parties or issuers themselves. MAOSCO is relatively open about its plans for future generations of the MULTOS standard. Published in late September 2000, MULTOS 5 adds various features including an optional dual interface (contact and contactless) to a single MULTOS chip, and support for GSM. Perhaps with the announcement of mobEcom's SecureSim on MULTOS, there will be a serious challenge to Java Card's dominance of the GSM market.

2.4.2.5 Comparison

The smallest code is that written in MEL for MULTOS. This is not surprising since the MULTOS virtual machine is optimised for smart card processors. Code generated from applications written in Java for Java Cards or C for MULTOS cards are around the same size, though usually the MULTOS is slightly smaller still. The code generated by the Microsoft VB compiler is much larger than for equivalent applications written for other MASCs. This seems to be largely due to compiler inefficiency and is likely to be fixed in the future.

WfSC execution speeds are not considered since the platform is not yet stable enough. The MULTOS code in MEL is fastest of the three platforms. MULTOS applications coded in C average 25% slower unless there are a lot of primitive calls, in which case there is no noticeable difference in speed. Java Card applications have been anything up to 50% slower than the equivalent MULTOS applications, though it has to be said that there are still many problems getting general applications to compile, load and execute on a range of Java Cards. It is interesting to note that Java Card SIMs are implemented as verified code rather than Java Card applets due to the performance issues with Java Card. In contrast, the SecureSIM on MULTOS from mobEcom is implemented in C as a MULTOS application that is executed in the virtual machine.

Each of the three MASC platforms has unique selling points. If you need interoperability with multiple sources of supply of known security level and with the ability to dynamically load and delete applications post issuance then MULTOS is the only choice. Java Cards compliant to JC 2.1.1 and OP 2.0 may emerge and begin to successfully interoperate over the next year or so. The remaining issue will still be their level of security. WfSC is not a mature offering at this point in time. Microsoft has the resources to move quickly and so WfSC may well soon be a serious contender. The three platforms will become more and more difficult to differentiate. For the time being, it is likely that the three will continue to operate in their own niches: Java Card largely in the mobile SIM world, MULTOS in long-term, high-volume rollouts where stability, known security and post issuance download are required and one day WfSC for use in applications in Windows environments [42].

2.4.3 Why Use Smart Cards?

The need to manage and secure a rapidly growing information network has focused increasing attention on smart card technology. Over the past decade, smart cards evolved from offering basic memory to complex systems with chips that incorporate powerful processing units with dedicated peripherals. This evolution enabled a wide range of applications. Smart card applications include financial transactions, e-commerce, physical access control, health and transportation services, and access to such wireless systems as the global system for mobile communication (GSM) and the upcoming universal mobile telecommunications system (UMTS) third-generation mobile phones [6].

Such applications depend on smart cards equipped to perform onboard cryptographic digital-signature encryption and authentication. Smart card operating systems use these cryptographic features to manage data storage and control access to private information. Essentially smart cards serve as security tokens by securely storing users' personal data and service providers' private information. The card interacts within a system using special communication interfaces and dedicated protocols. Smart cards provide highly reliable mechanisms for storing, accessing, and using data in non-volatile memory. Data access control and data management follow a security policy based on cryptographic service and defined for a specific application.

Sensitive data such as personal information, secret keys, and private application information stored in smart card memory is protected by combined hardware and software mechanisms. The write/store operation is more aggressively protected in smart cards than in any other device. Special onboard security sensors prevent alterations to memory during data storage or reading. In addition, the software includes a backup mechanism in case of card power-down during storage. Access and storage mechanisms can be combined of typical OS access management and systematic cryptographic verification to authenticate the application or to ensure transaction confidentiality. Smart card hardware features like a hardwired firewall between memory areas can make storage even more secure. Even if chemical or electrical corruption alters memory, hardware memory integrity checks or a software checksum will detect the alteration. Finally, hardware and software protect against illegal reading of smart card data. These combined measures offer powerful assurance of data privacy. Access management is far more secure in smart cards than in any computer OS. Objects, files, and keys can be protected during read, write, and execution

by secret codes or authentication-granted access rights set up by keys used in symmetric cryptography, such as DES, or asymmetric cryptography, such as RSA algorithms.

To ensure authentication, confidentiality, and integrity through cryptography, smart cards have enhanced arithmetic computation capabilities. Typical smart cards use secret-key algorithms such as the well-known Data Encryption Standard, the Advanced Encryption Standard, or other proprietary algorithms specified by operators. These algorithms mainly use data substitutions, permutations, compression and table lookups, and Boolean-to-arithmetic conversions. These generally simple operations lead to fast implementations even when performed in a high-level language. The associated keys are short (from 56 to 256 bits) and quite easy to manage. High-end smart cards offer far more powerful cryptographic algorithms, known as public-key algorithms. Examples include RSA for encryption/decryption and digital signatures. These schemes require an arithmetic unit to compute modular multiplication and reduction on large numbers, because the keys are at least 512 bits long and may reach 2,048 bits. With either type of algorithm, chips may have dedicated peripherals such as DES or RSA cryptoprocessors for efficiency. The OS would use such peripherals, for example, during the authentication scheme, either directly or by adding software to enhance hardware security [36].

Today's cards contain at least 128 Kbytes of ROM, associated with 64 to 128 Kbytes of EEPROM or flash memory, and 4 to 8 Kbytes of RAM. This compares with 16 Kbytes of ROM, 4 Kbytes of EEPROM, and 256 bytes for RAM offered a few years ago. Silicon technology enabled most of this progress by reducing the transistor scale for smart cards. To overcome external clock limitations, chips now run with their own internal clock, independent or not of the external one. Chips were using slow external clocks, even for heavy internal computations such as public-key cryptography [49]. Existing smart card terminals were not able to provide higher frequencies and modifying all the terminals was impractical. Chips with asynchronous communications provided the solution: The CPU runs its own clock and uses an external clock only for communication. CPUs and their peripherals can now run a 30-MHz internal clock, increasing chip speed by a factor of 6 to 10.